

CYLINDRICAL ALGEBRAIC DECOMPOSITION USING LOCAL PROJECTIONS

ADAM STRZEBONSKI

ABSTRACT. We present an algorithm which computes a cylindrical algebraic decomposition of a semialgebraic set using projection sets computed for each cell separately. Such local projection sets can be significantly smaller than the global projection set used by the Cylindrical Algebraic Decomposition (CAD) algorithm. This leads to reduction in the number of cells the algorithm needs to construct. We give an empirical comparison of our algorithm and the classical CAD algorithm.

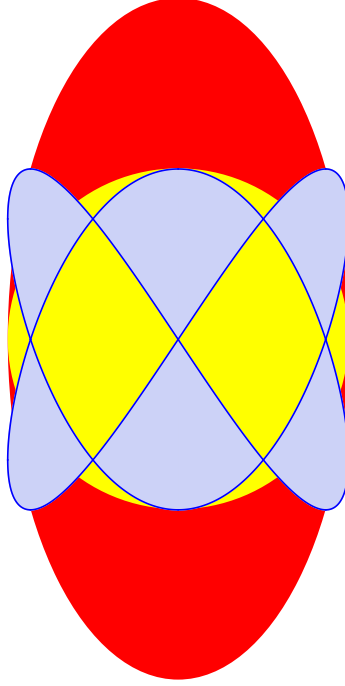
1. INTRODUCTION

A semialgebraic set is a subset of \mathbb{R}^n which is a solution set of a system of polynomial equations and inequalities. Computation with semialgebraic sets is one of the core subjects in computer algebra and real algebraic geometry. A variety of algorithms have been developed for real system solving, satisfiability checking, quantifier elimination, optimization and other basic problems concerning semialgebraic sets [7, 1, 5, 6, 9, 10, 12, 15, 18, 24, 25]. Every semialgebraic set can be represented as a finite union of disjoint cells bounded by graphs of algebraic functions. The Cylindrical Algebraic Decomposition (CAD) algorithm [7, 5, 21] can be used to compute a cell decomposition of any semialgebraic set presented by a quantified system of polynomial equations and inequalities. An alternative method of computing cell decompositions is given in [6]. Cell decompositions computed by the CAD algorithm can be represented directly [21, 22, 3] as cylindrical algebraic formulas (CAF; see the next section for a precise definition). A CAF representation of a semialgebraic set A can be used to decide whether A is nonempty, to find the minimal and maximal values of the first coordinate of elements of A , to generate an arbitrary element of A , to find a graphical representation of A , to compute the volume of A , or to compute multidimensional integrals over A (see [20]).

The CAD algorithm takes a system of polynomial equations and inequalities and constructs a cell decomposition of its solution set. The algorithm consists of two phases. The projection phase finds a set of polynomials whose roots are sufficient to describe the cell boundaries. The lifting phase constructs a cell decomposition, one dimension at a time, subdividing cells at all roots of the projection polynomials. However, some of these subdivisions may be unnecessary, either because of the geometry of the roots or because of the Boolean structure of the input system. In this paper we propose an algorithm which combines the two phases. It starts with a sample point and constructs a cell containing the point on which the input system has a constant truth value. Projection polynomials used to construct the cell are selected based on the structure of the system at the sample point. Such a local projection set can often be much smaller than the global projection set used by the CAD algorithm. The idea to use such locally valid projections was first introduced in [13], in an algorithm to decide the satisfiability of systems of real polynomial equations

and inequalities. It was also used in [4], in an algorithm to construct a single open cell from a cylindrical algebraic decomposition.

Example 1. Find a cylindrical algebraic decomposition of the solution set of $S = f_1 < 0 \vee (f_2 \leq 0 \wedge f_3 \leq 0)$, where $f_1 = 4x^2 + y^2 - 4$, $f_2 = x^2 + y^2 - 1$, and $f_3 = 16x^6 - 24x^4 + 9x^2 + 4y^4 - 4y^2$.



The solution set of S is equal to the union of the open ellipse $f_1 < 0$ and the intersection of the closed disk $f_2 \leq 0$ and the set $f_3 \leq 0$ bounded by a Lissajous curve. As can be seen in the picture, the set is equal to the open ellipse $f_1 < 0$. The CAD algorithm uses a projection set consisting of the discriminants and the pairwise resultants of f_1 , f_2 , and f_3 . It computes a cell decomposition of the solution set of S by constructing 357 cells such that all f_1 , f_2 , and f_3 have a constant sign on each cell. Note however, that a cell decomposition of the solution set of S can be obtained by considering the following 13 cells. On each cell only some of f_1 , f_2 , and f_3 have a constant sign, but those signs are sufficient to determine the truth value of S .

- (1) S is true on $-1 < x < 1 \wedge -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}$ because $f_1 < 0$.
- (2) S is false on $-1 < x < 1 \wedge y < -2\sqrt{1-x^2}$ and on $-1 < x < 1 \wedge y > 2\sqrt{1-x^2}$ because $f_1 > 0 \wedge f_2 > 0$.
- (3) S is false on $-1 < x < 1 \wedge y = -2\sqrt{1-x^2}$ and on $-1 < x < 1 \wedge y = 2\sqrt{1-x^2}$ because $f_1 = 0 \wedge f_2 > 0$.
- (4) S is false on $x < -1$ and on $x > 1$ because $f_1 > 0 \wedge f_2 > 0$.
- (5) S is false on $x = -1 \wedge y < 0$ and on $x = -1 \wedge y > 0$ because $f_1 > 0 \wedge f_2 > 0$.
- (6) S is false on $x = -1 \wedge y = 0$ because $f_1 = 0 \wedge f_3 > 0$.
- (7) S is false on $x = 1 \wedge y < 0$ and on $x = 1 \wedge y > 0$ because $f_1 > 0 \wedge f_2 > 0$.
- (8) S is false on $x = 1 \wedge y = 0$ because $f_1 = 0 \wedge f_3 > 0$.

Determining the cell bounds for the cell stack (1)-(3) requires computation of roots of $\text{discr}_y f_1$, $\text{discr}_y f_2$, and $\text{res}_y(f_1, f_2)$ in x and roots of $f_1(0, y)$ and $f_2(0, y)$ in y . Determining the cell bounds for the cells (4) requires computation of roots of $\text{discr}_y f_1$ and $\text{discr}_y f_2$ in x and roots of $f_1(-2, y)$, $f_2(-2, y)$, $f_1(2, y)$ and $f_2(2, y)$ in y . Determining the cell bounds for the cell stacks (5)-(6) and (7)-(8) requires computation of roots of $f_1(-1, y)$, $f_2(-1, y)$, $f_3(-1, y)$, $f_1(1, y)$, $f_2(1, y)$ and $f_3(1, y)$ in y . Polynomial f_3 is not used to compute any of the projections and its roots in y are computed only for two values of x . The algorithm we propose in this paper computes a cell decomposition of the solution set of S by constructing the 13 cells given in (1)-(8). Details of the computation for this example are given in Section 3.5.

Example 2. Find a cylindrical algebraic decomposition of the solution set of $S = ax^4 + bx^3 + cx^2 + dx + e \geq 0$ in the variable order (a, b, c, d, e, x) .

In this example the system is not well-oriented, hence the CAD algorithm needs to use Hong's projection operator for the first three projections. However, the additional projection polynomials are necessary only for the cells on which a McCallum's projection polynomial vanishes identically. For most cells local projection can be computed using McCallum's projection operator, and for the few cells on which a McCallum's projection polynomial vanishes identically local projection needs to use some, but usually not all, polynomials from Hong's projection operator. The algorithm LPCAD we propose in this paper computes a cell decomposition of the solution set of S by constructing 523 cells in 0.95 seconds of CPU time. The CAD algorithm did not finish the computation in 72 hours. A version of LPCAD using only local projections based on Hong's projection operator constructs 1375 cells and takes 2.72 seconds of CPU time.

2. PRELIMINARIES

A system of polynomial equations and inequalities in variables x_1, \dots, x_n is a formula

$$S(x_1, \dots, x_n) = \bigvee_{1 \leq i \leq l} \bigwedge_{1 \leq j \leq m} f_{i,j}(x_1, \dots, x_n) \rho_{i,j} 0$$

where $f_{i,j} \in \mathbb{R}[x_1, \dots, x_n]$, and each $\rho_{i,j}$ is one of $<, \leq, \geq, >, =$, or \neq .

A subset of \mathbb{R}^n is *semialgebraic* if it is a solution set of a system of polynomial equations and inequalities.

A *quantified system of real polynomial equations and inequalities* in free variables x_1, \dots, x_n and quantified variables t_1, \dots, t_m is a formula

$$Q_1 t_1 \dots Q_m t_m S(t_1, \dots, t_m; x_1, \dots, x_n)$$

Where Q_i is \exists or \forall , and S is a system of real polynomial equations and inequalities in $t_1, \dots, t_m, x_1, \dots, x_n$.

By Tarski's theorem (see [24]), solution sets of quantified systems of real polynomial equations and inequalities are semialgebraic.

Notation 3. For $k \geq 1$, let \bar{a} denote a k -tuple (a_1, \dots, a_k) of real numbers and let \bar{x} denote a k -tuple (x_1, \dots, x_k) of variables.

Every semialgebraic set can be represented as a finite union of disjoint *cells* (see [14]), defined recursively as follows.

- (1) A cell in \mathbb{R} is a point or an open interval.

(2) A cell in \mathbb{R}^{k+1} has one of the two forms

$$\begin{aligned} & \{(\bar{a}, a_{k+1}) : \bar{a} \in C_k \wedge a_{k+1} = r(\bar{a})\} \\ & \{(\bar{a}, a_{k+1}) : \bar{a} \in C_k \wedge r_1(\bar{a}) < a_{k+1} < r_2(\bar{a})\} \end{aligned}$$

where C_k is a cell in \mathbb{R}^k , r is a continuous algebraic function, and r_1 and r_2 are continuous algebraic functions, $-\infty$, or ∞ , and $r_1 < r_2$ on C_k .

A finite collection D of cells in \mathbb{R}^n is *cylindrically arranged* if for any $C_1, C_2 \in D$ and $k \leq n$ the projections of C_1 and C_2 on \mathbb{R}^k are either disjoint or identical.

Given a semialgebraic set presented by a quantified system of polynomial equations and inequalities, the CAD algorithm can be used to decompose the set into a cylindrically arranged finite collection of cells. The collection of cells is represented by a cylindrical algebraic formula (CAF). A CAF describes each cell by giving explicit algebraic function bounds and the Boolean structure of a CAF reflects the cylindrical arrangement of cells. Before we give a formal definition of a CAF, let us first introduce some terminology.

Let $k \geq 1$ and let $f = c_d y^d + \dots + c_0$, where $c_0, \dots, c_d \in \mathbb{Z}[\bar{x}]$. A *real algebraic function* given by the *defining polynomial* f and a *root number* $p \in \mathbb{N}_+$ is the function

$$(2.1) \quad \text{Root}_{y,p}f : \mathbb{R}^k \ni \bar{a} \longrightarrow \text{Root}_{y,p}f(\bar{a}) \in \mathbb{R}$$

where $\text{Root}_{y,p}f(\bar{a})$ is the p -th real root of $f(\bar{a}, y) \in \mathbb{R}[y]$. The function is defined for those values of \bar{a} for which $f(\bar{a}, y)$ has at least p real roots. The real roots are ordered by the increasing value and counted with multiplicities. A real algebraic number $\text{Root}_{y,p}f \in \mathbb{R}$ given by a *defining polynomial* $f \in \mathbb{Z}[y]$ and a *root number* p is the p -th real root of f . See [19, 20] for more details on how algebraic numbers and functions can be implemented in a computer algebra system.

Let C be a connected subset of \mathbb{R}^k . $\text{Root}_{y,p}f$ is *regular* on C if it is continuous on C , $c_d(\bar{a}) \neq 0$ for all $\bar{a} \in C$, and there exist $m \in \mathbb{N}_+$ such that for any $\bar{a} \in C$ $\text{Root}_{y,p}f(\bar{a})$ is a root of $f(\bar{a}, y)$ of multiplicity m .

f is *degree-invariant* on C if there exist $e \in \mathbb{N}$ such that if $c_d(\bar{a}) = \dots = c_{e+1}(\bar{a}) = 0 \wedge c_e(\bar{a}) \neq 0$ for all $\bar{a} \in C$.

A set $W = \{f_1, \dots, f_m\}$ of polynomials is *delineable* on C if all elements of W are degree-invariant on C and for $1 \leq i \leq m$

$$f_i^{-1}(0) \cap (C \times \mathbb{R}) = \{r_{i,1}, \dots, r_{i,l_i}\}$$

where $r_{i,1}, \dots, r_{i,l_i}$ are disjoint regular real algebraic functions and for $i_1 \neq i_2$ r_{i_1,j_1} and r_{i_2,j_2} are either disjoint or equal. Functions $r_{i,j}$ are *root functions of f_i over C* .

A set $W = \{f_1, \dots, f_m\}$ of polynomials is *analytic delineable* on a connected analytic submanifold C of \mathbb{R}^k if W is delineable on C and the root functions of elements of W over C are analytic.

Let W be delineable on C , let $r_1 < \dots < r_l$ be all root functions of elements of W over C , and let $r_0 = -\infty$ and $r_{l+1} = \infty$. For $1 \leq i \leq l$, the i -th W -section over C is the set

$$\{(\bar{a}, a_{k+1}) : \bar{a} \in C \wedge a_{k+1} = r_i(\bar{a})\}$$

For $1 \leq i \leq l+1$, the i -th W -sector over C is the set

$$\{(\bar{a}, a_{k+1}) : \bar{a} \in C \wedge r_{i-1}(\bar{a}) < a_{k+1} < r_i(\bar{a})\}$$

A formula F is an *algebraic constraint with bounds* $BDS(F)$ if it is a level- k equational or inequality constraint with $1 \leq k \leq n$ defined as follows.

- (1) A *level-1 equational constraint* has the form $x_1 = r$, where r is a real algebraic number, and $BDS(F) = \{r\}$.

- (2) A level-1 inequality constraint has the form $r_1 < x_1 < r_2$, where r_1 and r_2 are real algebraic numbers, $-\infty$, or ∞ , and $BDS(F) = \{r_1, r_2\} \setminus \{-\infty, \infty\}$.
- (3) A level- $k+1$ equational constraint has the form $x_{k+1} = r(\bar{x})$, where r is a real algebraic function, and $BDS(F) = \{r\}$.
- (4) A level- $k+1$ inequality constraint has the form $r_1(\bar{x}) < x_{k+1} < r_2(\bar{x})$, where r_1 and r_2 are real algebraic functions, $-\infty$, or ∞ , and $BDS(F) = \{r_1, r_2\} \setminus \{-\infty, \infty\}$.

A level- $k+1$ algebraic constraint F is *regular* on a connected set $C \subseteq \mathbb{R}^k$ if all elements of $BDS(F)$ are regular on C and, if F is an inequality constraint, $r_1 < r_2$ on C .

Definition 4. An atomic cylindrical algebraic formula (CAF) F in (x_1, \dots, x_n) has the form $F_1 \wedge \dots \wedge F_n$, where F_k is a level- k algebraic constraint for $1 \leq k \leq n$ and F_{k+1} is regular on the solution set of $F_1 \wedge \dots \wedge F_k$ for $1 \leq k < n$.

Level- k cylindrical subformulas are defined recursively as follows

- (1) A level- n cylindrical subformula is a disjunction of level- n algebraic constraints.
- (2) A level- k cylindrical subformula, with $1 \leq k < n$, has the form

$$(F_1 \wedge G_1) \vee \dots \vee (F_m \wedge G_m)$$

where F_i are level- k algebraic constraints and G_i are level- $k+1$ cylindrical subformulas.

A cylindrical algebraic formula (CAF) is a level-1 cylindrical subformula F such that distributing conjunction over disjunction in F gives

$$DNF(F) = F_1 \vee \dots \vee F_l$$

where each F_i is an atomic CAF.

Given a quantified system of real polynomial equations and inequalities the CAD algorithm [21] returns a CAF representation of its solution set.

Example 5. The following formula $F(x, y, z)$ is a CAF representation of the closed unit ball.

$$\begin{aligned}
 F(x, y, z) &:= x = -1 \wedge y = 0 \wedge z = 0 \vee \\
 &\quad -1 < x < 1 \wedge b_2(x, y, z) \vee \\
 &\quad x = 1 \wedge y = 0 \wedge z = 0 \\
 b_2(x, y, z) &:= y = R_1(x) \wedge z = 0 \vee \\
 &\quad R_1(x) < y < R_2(x) \wedge b_{2,2}(x, y, z) \vee \\
 &\quad y = R_2(x) \wedge z = 0 \\
 b_{2,2}(x, y, z) &:= z = R_3(x, y) \vee \\
 &\quad R_3(x, y) < z < R_4(x, y) \vee \\
 &\quad z = R_4(x, y)
 \end{aligned}$$

where

$$\begin{aligned}
 R_1(x) &= \text{Root}_{y,1}(x^2 + y^2) = -\sqrt{1-x^2} \\
 R_2(x) &= \text{Root}_{y,2}(x^2 + y^2) = \sqrt{1-x^2} \\
 R_3(x, y) &= \text{Root}_{z,1}(x^2 + y^2 + z^2) = -\sqrt{1-x^2-y^2} \\
 R_4(x, y) &= \text{Root}_{z,2}(x^2 + y^2 + z^2) = \sqrt{1-x^2-y^2}
 \end{aligned}$$

3. CAD CONSTRUCTION USING LOCAL PROJECTIONS

In this section we describe an algorithm for computing a CAF representation of the solution set of a system of polynomial equations and inequalities. The algorithm uses local projections computed separately for each cell. For simplicity we assume that the system is not quantified. The algorithm can be extended to quantified systems following the ideas of [8]. The algorithm in its version given here does not take advantage of equational constraints. The use of equational constraints will be described in the full version of the paper.

The main, recursive, algorithm used for CAD construction is Algorithm 13. Let us sketch the algorithm here, a detailed description is given later in this section. The input is a system $S(x_1, \dots, x_n)$ of polynomial equations and inequalities and a point $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$ with $0 \leq k < n$. The algorithm finds a level- $k+1$ cylindrical subformula F and a set of polynomials $V \subseteq \mathbb{R}[x_1, \dots, x_k]$ such that for any cell $C \subseteq \mathbb{R}^k$ containing \bar{a} on which all elements of V have constant signs

$$(x_1, \dots, x_k) \in C \Rightarrow (F(x_1, \dots, x_n) \iff S(x_1, \dots, x_n))$$

The formula F can be interpreted as a description of the solution set of S as a finite collection of cylindrically arranged cells in \mathbb{R}^{n-k} , parametrized by the values of (x_1, \dots, x_k) . The description is valid locally to \bar{a} , where the meaning of “locally” is determined by V . The approach is to find algebraic constraints

$$G_1(\bar{x}, x_{k+1}), \dots, G_m(\bar{x}, x_{k+1})$$

and cylindrical subformulas H_1, \dots, H_m such that the solution sets of

$$G_1(\bar{a}, x_{k+1}), \dots, G_m(\bar{a}, x_{k+1})$$

form a decomposition of \mathbb{R} and H_i describes the solution set of S locally to $\{\bar{a}\} \times \{x_{k+1} : G_i(\bar{a}, x_{k+1})\}$. To find G 's, H 's, and V we start with a stack containing the interval $(-\infty, \infty)$ and until the stack is emptied execute the following steps. We take an interval I off stack and pick $a_{k+1} \in I$. If evaluating the $k+1$ -variate polynomials in S at (\bar{a}, a_{k+1}) suffices to establish the truth value of S , let P be a set of $k+1$ -variate polynomials in S sufficient to establish the truth value of S and let H be the truth value. Otherwise, let H and P be, respectively, the formula and the set of polynomials returned by Algorithm 13 applied to S and (\bar{a}, a_{k+1}) . We use projection to compute a set $W \subseteq \mathbb{R}[x_1, \dots, x_k]$ such that P is delineable on any cell containing \bar{a} on which all elements of W have constant signs and we add the elements of W to V . Let J be the interval containing a_{k+1} bounded by the nearest roots of elements of P and let G be the constraint on x_{k+1} whose bounds are the corresponding algebraic functions. Note that if P is delineable on a cell C containing \bar{a} then the elements of P have constant signs on $D = \{(\bar{x}, x_{k+1}) : \bar{x} \in C \wedge G(\bar{x}, x_{k+1})\}$ and hence H is equivalent to S on D . We add G and H to the list of G 's, H 's, and, if $I \setminus J$ is nonempty, we add the components of $I \setminus J$ to stack. When the stack is empty we use projection to compute a set $W \subseteq \mathbb{R}[x_1, \dots, x_k]$ such the set of polynomials whose roots appear as bounds in G 's are delineable on any cell containing \bar{a} on which all elements of W have constant signs and we add the elements of W to V . As required, the formula $F = (G_1 \wedge H_1) \vee \dots \vee (G_m \wedge H_m)$ is equivalent to S on any cell containing \bar{a} on which all elements of V have constant signs.

To compute a CAF representation of the solution set of S we call Algorithm 13 with $k = 0$.

Notation 6. *We will use the following notations.*

- (1) For a finite set of polynomials P , let \bar{P} denote the set of irreducible factors of the elements of P .
- (2) Let IRR_k denote the irreducible elements of $\mathbb{R}[x_1, \dots, x_k] \setminus \mathbb{R}[x_1, \dots, x_{k-1}]$.
- (3) For a set $A \subseteq \mathbb{R}^n$ and $k \leq n$ let $\Pi_k(A)$ denote the projection of A on \mathbb{R}^k .

In this section we assume that all polynomials have coefficients in a fixed computable subfield $K \subseteq \mathbb{R}$, irreducibility is understood to be in the ring of polynomials with coefficients in K , irreducible factors are always content-free and chosen in a canonical way, and finite sets of polynomials are always ordered according to a fixed linear ordering in the set of all polynomials with coefficients in K . In our implementation $K = \mathbb{Q}$.

Whenever we write $a = (a_1, \dots, a_k) \in \mathbb{R}^k$ with $k \geq 0$ we include the possibility of $a = ()$, the only element of \mathbb{R}^0 .

3.1. Local projection.

Definition 7. Let $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ be a finite set of polynomials and let $a = (a_1, \dots, a_{n-1}) \in \mathbb{R}^{n-1}$, where $n \geq 1$. Let $W = (W_1, \dots, W_n)$ be such that W_k is a finite subset of IRR_k and $\bar{P} \cap IRR_k \subseteq W_k$ for $1 \leq k \leq n$. W is a local projection sequence for P at a iff, for any $1 \leq k < n$ and any cell $C \subseteq \mathbb{R}^k$, if $(a_1, \dots, a_k) \in C$ and all elements of W_j for $1 \leq j \leq k$ have constant signs on $\Pi_j(C)$ then the set of elements of W_{k+1} that are not identically zero on $C \times \mathbb{R}$ is delineable over C .

To compute local projections we use the following two projection procedures, derived, respectively, from McCallum's projection operator [16, 17, 2] and Hong's projection operator [11].

Algorithm 8. (LProjMC)

Input: $P = \{p_1, \dots, p_m\} \subseteq IRR_{k+1}$ and $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$, where $k \geq 1$.

Output: A finite set $Q \subseteq \mathbb{R}[x_1, \dots, x_k]$.

(1) Put $Q = \emptyset$ and compute $R = \{p \in P : \exists b \in \mathbb{R} p(\bar{a}, b) = 0\}$.

(2) For $1 \leq i \leq m$ do

(a) Let $p_i = q_d x_{k+1}^d + \dots + q_0$. Put $Q = Q \cup \{q_d\}$.

(b) If $k > 1$ and $q_d(\bar{a}) = \dots = q_0(\bar{a}) = 0$ put

$$Q = Q \cup \{q_{d-1}, \dots, q_0\}$$

and continue the loop.

(c) If $k > 1$, $q_d(\bar{a}) = 0$, and none of q_{d-1}, \dots, q_0 is a nonzero constant, put $Q = Q \cup \{q_l\}$, where l is maximal such that $q_l(\bar{a}) \neq 0$.

(d) Put $Q = Q \cup \{disc_{x_{k+1}} p_i\}$.

(e) If $p_i \in R$ then put

$$Q = Q \cup \{res_{x_{k+1}}(p_i, p_j) : i < j \leq m \wedge p_j \in R\}$$

(3) Return Q .

In the next algorithm we use the following notation.

Notation 9. Let $f, g \in \mathbb{R}[\bar{x}][x_{k+1}]$, $\bar{a} \in \mathbb{R}^k$, and

$$d = \min(\deg(f), \deg(g))$$

If for some $0 \leq l < d$, $psc_0(f, g)(\bar{a}) = \dots = psc_{l-1}(f, g)(\bar{a}) = 0$ and $psc_l(f, g)(\bar{a}) \neq 0$, then $PSC(f, g, \bar{a}) := \{psc_0(f, g), \dots, psc_l(f, g)\}$. Otherwise

$$PSC(f, g, \bar{a}) := \{psc_0(f, g), \dots, psc_{d-1}(f, g)\}$$

Algorithm 10. (*LProjH*)

Input: $P = \{p_1, \dots, p_m\} \subseteq \text{IRR}_{k+1}$ and $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$, where $k \geq 1$.

Output: A finite set $Q \subseteq \mathbb{R}[x_1, \dots, x_k]$.

- (1) Put $Q = \emptyset$ and compute $R = \{p \in P : \exists b \in \mathbb{R} p(\bar{a}, b) = 0\}$.
- (2) For $1 \leq i \leq m$ do
 - (a) Let $p_i = q_d x_{k+1}^d + \dots + q_0$. Put $Q = Q \cup \{q_d\}$ and $r_i = p_i$.
 - (b) If $q_d(\bar{a}) = \dots = q_0(\bar{a}) = 0$ put $Q = Q \cup \{q_{d-1}, \dots, q_0\}$ and continue the loop.
 - (c) If $q_d(\bar{a}) = 0$, put $Q = Q \cup \{q_{d-1}, \dots, q_l\}$ and $r_i = q_l x_{k+1}^l + \dots + q_0$, where l is maximal such that $q_l(\bar{a}) \neq 0$.
 - (d) Put $Q = Q \cup \text{PSC}(r_i, \frac{\partial r_i}{\partial x_{k+1}}, \bar{a})$.
 - (e) If $p_i \in R$ then for $i < j \leq m$ if $p_j \in R$ put $Q = Q \cup \text{PSC}(r_i, p_j, \bar{a})$.
- (3) Return Q .

The following algorithm computes a local projection for given P and a .

Algorithm 11. (*LocalProjection*)

Input: A finite set $P \subseteq \mathbb{R}[x_1, \dots, x_n]$ and $a = (a_1, \dots, a_{n-1}) \in \mathbb{R}^{n-1}$, where $n \geq 1$.

Output: A local projection sequence $W = (W_1, \dots, W_n)$ for P at a .

- (1) Set $wo = \text{true}$, $Q = P$, $k = n - 1$.
- (2) While $k \geq 1$ do
 - (a) Let $\bar{a} = (a_1, \dots, a_k)$ and compute $W_{k+1} = \bar{Q} \cap \text{IRR}_{k+1}$, $Q = \bar{Q} \setminus W_{k+1}$.
 - (b) If $wo = \text{true}$, $1 < k < n - 1$, and an element of W_{k+1} is identically zero at \bar{a} , then set $wo = \text{false}$, $Q = P$, $k = n - 1$ and continue the loop.
 - (c) If $wo = \text{true}$ or $k \leq 2$ set $Q = Q \cup \text{LProjMC}(W_{k+1}, \bar{a})$ else set $Q = Q \cup \text{LProjH}(W_{k+1}, \bar{a})$.
 - (d) Set $k = k - 1$.
- (3) Set $W_1 = \bar{Q} \cap \text{IRR}_1$.
- (4) Return $W = (W_1, \dots, W_n)$.

3.2. The CAD construction algorithm. Let us first introduce an algorithm for evaluation of polynomial systems at “partial” sample points.

Algorithm 12. (*PEval*)

Input: A system $S(x_1, \dots, x_n)$ of polynomial equations and inequalities and $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$ with $0 \leq k \leq n$.

Output: *undecided* or a pair (v, P) , where $v \in \{\text{true}, \text{false}\}$, $P = \{p_1, \dots, p_m\} \subseteq \mathbb{R}[x_1, \dots, x_k]$, and for any $b = (b_1, \dots, b_n) \in \mathbb{R}^n$ if

$$\text{sign}(p_i(a_1, \dots, a_k)) = \text{sign}(p_i(b_1, \dots, b_k))$$

for all $1 \leq i \leq m$ then the value of $S(b)$ is v .

- (1) If $S = \text{false}$ or $S = \text{true}$ then return (S, \emptyset) .
- (2) If $S = (f \rho 0)$, where ρ is one of $<, \leq, \geq, >, =, \text{ or } \neq$.
 - (a) If there exists a factor g of f such that $g \in \mathbb{R}[x_1, \dots, x_k]$ and $g(\bar{a}) = 0$ then return $(0 \rho 0, \{g\})$.
 - (b) If $f \in \mathbb{R}[x_1, \dots, x_k]$ return $(f(\bar{a}) \rho 0, \{f\})$.
 - (c) Return *undecided*.
- (3) If $S = T_1 \wedge \dots \wedge T_l$
 - (a) For $1 \leq i \leq l$ compute $e_i = \text{PEval}(T_i, \bar{a})$.
 - (b) If for some i $e_i = (\text{false}, P_i)$ then return (false, P_i) .
 - (c) If for all i $e_i = (\text{true}, P_i)$ then return $(\text{true}, P_1 \cup \dots \cup P_l)$.

- (d) *Return undecided.*
- (4) *If $S = T_1 \vee \dots \vee T_l$*
 - (a) *For $1 \leq i \leq l$ compute $e_i = PEval(T_i, \bar{a})$.*
 - (b) *If for some i $e_i = (true, P_i)$ then return $(true, P_i)$.*
 - (c) *If for all i $e_i = (false, P_i)$ then return $(false, P_1 \cup \dots \cup P_l)$.*
 - (d) *Return undecided.*

We can now present a recursive algorithm computing cylindrical algebraic decomposition using local projections.

Algorithm 13. (LPCAD)

Input: A system $S(x_1, \dots, x_n)$ of polynomial equations and inequalities and $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$ with $0 \leq k < n$.

Output: A pair (F, V) , where F is a level- $k+1$ cylindrical subformula, $V = (V_1, \dots, V_k)$, $V_j \subseteq \mathbb{R}[x_1, \dots, x_j]$ for $1 \leq j \leq k$, and for any cell $C \subseteq \mathbb{R}^k$ if $\bar{a} \in C$ and for $1 \leq j \leq k$ all elements of V_j have constant signs on $\Pi_j(C)$ then

$$(x_1, \dots, x_k) \in C \Rightarrow (F(x_1, \dots, x_n) \iff S(x_1, \dots, x_n))$$

- (1) *Compute a disjunctive normal form S_{DNF} and a conjunctive normal form S_{CNF} of S .*
- (2) *Set $stack = \{(-\infty, -\infty, <, \infty, \infty, <)\}$ and $A = Q = V_1 = \dots = V_k = \emptyset$.*
- (3) *While $stack \neq \emptyset$ do*
 - (a) *Remove a tuple $(u_1, r_1, \rho_1, u_2, r_2, \rho_2)$ from $stack$. r_1, r_2 are algebraic functions of x_1, \dots, x_k , $-\infty$, or ∞ , $u_1 = r_1(\bar{a})$, $u_2 = r_2(\bar{a})$, $\rho_1, \rho_2 \in \{<, \leq\}$, and the tuple represents the interval $u_1 \rho_1 x_{k+1} \rho_2 u_2$.*
 - (b) *If $u_1 = u_2$ set $a_{k+1} = u_1$ and set $R = \{f\}$, where $r_1 = Root_{x_{k+1}, p} f$, else pick a rational number $u_1 < a_{k+1} < u_2$ and set $R = \emptyset$. Set $\bar{b} = (\bar{a}, a_{k+1})$.*
 - (c) *Compute $e_{CNF} = PEval(S_{CNF}, \bar{b})$. If $e_{CNF} = (false, P)$ then set $H = false$ and $W = LocalProjection(P \cup R, \bar{a})$, and go to (f).*
 - (d) *Compute $e_{DNF} = PEval(S_{DNF}, \bar{b})$. If $e_{DNF} = (true, P)$ then set $H = true$ and $W = LocalProjection(P \cup R, \bar{a})$, and go to (f).*
 - (e) *Compute $(H, U) = LPCAD(S, \bar{b})$. For $1 \leq j \leq k$ set $V_j = V_j \cup U_j$. Compute $W = LocalProjection(U_{k+1} \cup R, \bar{a})$.*
 - (f) *For $1 \leq j \leq k$ set $V_j = V_j \cup W_j$.*
 - (g) *If $u_1 = u_2$ then set $G = (x_{k+1} = r_1)$ and go to (n).*
 - (h) *Find s_1 and s_2 such that*
 - (i) $s_1 = Root_{x_{k+1}, p_1} f_1$ and $f_1 \in W_{k+1}$ or $s_1 = f_1 \equiv -\infty$,
 - (ii) $s_2 = Root_{x_{k+1}, p_2} f_2$ and $f_2 \in W_{k+1}$ or $s_2 = f_2 \equiv \infty$,
 - (iii) $v_1 = s_1(\bar{a})$ and $v_2 = s_2(\bar{a})$,
 - (iv) *either $v_1 = v_2 = a_{k+1}$ or $v_1 < a_{k+1} < v_2$ and there are no roots of elements of W_{k+1} in (v_1, v_2) .*
 - (i) *Set $Q = Q \cup (\{f_1, f_2\} \setminus \{-\infty, \infty\})$.*
 - (j) *If $v_1 = v_2$ then set $G = (x_{k+1} = s_1)$, add*

$$(v_1, s_1, <, u_2, r_2, \rho_2)$$

and

$$(u_1, r_1, \rho_1, v_1, s_1, <)$$

to $stack$, and go to (n).

- (k) If $u_2 < v_2$ then set $t_2 = r_2$ and $\sigma_2 = \rho_2$. Else set $t_2 = s_2$ and $\sigma_2 = <$, and if $u_2 > v_2$ or $\rho_2 = \leq$ add

$$(v_2, s_2, \leq, u_2, r_2, \rho_2)$$

to stack.

- (l) If $v_1 < u_1$ then set $t_1 = r_1$ and $\sigma_1 = \rho_1$. Else set $t_1 = s_1$ and $\sigma_1 = <$, and if $v_1 > u_1$ or $\rho_1 = \leq$ add

$$(u_1, r_1, \rho_1, v_1, s_1, \leq)$$

to stack.

- (m) Set $G = (t_1 \sigma_1 x_{k+1} \sigma_2 t_2)$.

- (n) Set $A = A \cup \{(a_{k+1}, G \wedge H)\}$

- (4) Sort A by increasing values of the first element, obtaining $\{(c_1, H_1), \dots, (c_m, H_m)\}$. Set $F = H_1 \vee \dots \vee H_m$.

- (5) Compute $W = \text{LocalProjection}(Q, \bar{a})$.

- (6) For $1 \leq j \leq k$ set $V_j = V_j \cup W_j$.

- (7) Return (F, V) .

Corollary 14. $\text{LPCAD}(S(x_1, \dots, x_n), ())$ returns

$$(F(x_1, \dots, x_n), ())$$

where $F(x_1, \dots, x_n)$ is a cylindrical algebraic formula equivalent to $S(x_1, \dots, x_n)$.

The formula returned by Algorithm 13 may involve weak inequalities, but it can be easily converted to the CAF format by replacing weak inequalities with disjunctions of equations and strict inequalities.

3.3. Proofs. To prove correctness of Algorithm 11 we use the following lemmata.

Lemma 15. Let $k \geq 1$, $P \subseteq \text{IRR}_{k+1}$, $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$, and $Q = \text{LProjMC}(P, \bar{a})$. If D is a connected analytic submanifold of \mathbb{R}^k such that $\bar{a} \in D$ and all elements of Q are order-invariant in D then the set P^* of all elements of P that are not identically zero on $D \times \mathbb{R}$ is analytic delineable over D and the elements of P^* are order-invariant in each P^* -section over D .

Proof. Suppose that $f \in P^*$. Step (2a) of Algorithm 8 guarantees that f has a sign-invariant leading coefficient in D . f does not vanish identically at any point in D (for $k > 1$ it is ensured by step (2c); for $k = 1$ it follows from irreducibility of f). By Theorem 3.1 of [2], f is degree-invariant on D . Since $\text{disc}_{x_{k+1}}(f) \in Q$, by Theorem 2 of [17], $\{f\}$ is analytic delineable over D and is order-invariant in each $\{f\}$ -section over D . Suppose that $g \in P^*$ and $g \neq f$. If either $f(\bar{a}, x_{k+1})$ or $g(\bar{a}, x_{k+1})$ has no real roots then $\{f, g\}$ is delineable on D . Otherwise $\text{res}_{x_{k+1}}(f, g) \in Q$ and hence, by Theorem 2 of [17], $\{f, g\}$ is analytic delineable over D . Therefore, P^* is analytic delineable over D and the elements of P^* are order-invariant in each P^* -section over D . \square

Lemma 16. Let $k \geq 1$, $P \subseteq \text{IRR}_{k+1}$, $\bar{a} = (a_1, \dots, a_k) \in \mathbb{R}^k$, and $Q = \text{LProjH}(P, \bar{a})$. If D is a connected subset of \mathbb{R}^k such that $\bar{a} \in D$ and all elements of Q are sign-invariant in D then the set P^* of all elements of P that are not identically zero on $D \times \mathbb{R}$ is delineable over D .

Proof. Suppose that $f = q_d x_{k+1}^d + \dots + q_0 \in P^*$. Let l be maximal such that $q_l(\bar{a}) \neq 0$, and let $f_{\text{red}} = q_l x_{k+1}^l + \dots + q_0$. Steps (2a) and (2c) of Algorithm 10 guarantee that $f = f_{\text{red}}$ in $D \times \mathbb{R}$. By step (2d) and Theorems 1-3 of [7], $\{f_{\text{red}}\}$ is delineable over D , and hence

$\{f\}$ is delineable over D . Suppose that $g \in P^*$ and $g \neq f$. If either $f(\bar{a}, x_{k+1})$ or $g(\bar{a}, x_{k+1})$ has no real roots then $\{f, g\}$ is delineable on D . Otherwise without loss of generality we may assume that due to step (2e) Q contains all factors of $PSC(f_{red}, g, \bar{a})$. By Lemma 1 of [11] and Theorem 2 of [7], the degree of $\gcd(f(\bar{b}, x_{k+1}), g(\bar{b}, x_{k+1}))$ is constant for $\bar{b} \in D$. Since f and g are degree-invariant in D , by Lemma 12 of [23], $\{f, g\}$ is delineable over D . Therefore P^* is delineable over D . \square

Proposition 17. *Algorithm 11 terminates and returns a local projection sequence for P at a .*

Proof. To show that the algorithm terminates note that the body of the loop in step (2) is executed at most $2n - 2$ times.

Let $W = (W_1, \dots, W_n)$ be the returned sequence. Steps (2a) and (3) ensure that W_k is a finite subset of IRR_k and $\bar{P} \cap IRR_k \subseteq W_k$ for $1 \leq k \leq n$. We will recursively construct a cell $D \subseteq \mathbb{R}^{n-1}$ such that $D_k = \Pi_k(D)$ is the maximal connected set containing $\Pi_k(a)$ such that all elements of W_j for $1 \leq j \leq k$ have constant signs on $\Pi_j(D_k)$. Moreover, for $1 \leq k < n$, the set W_{k+1}^* of elements of W_{k+1} that are not identically zero on $D_k \times \mathbb{R}$ is delineable over D_k . This is sufficient to prove that W is a local projection sequence for P at a , because for any cell $C \subseteq \mathbb{R}^k$ if $(a_1, \dots, a_k) \in C$ and all elements of W_j for $1 \leq j \leq k$ have constant signs on $\Pi_j(C)$ then $C \subseteq D_k$, by maximality of D_k .

We will consider two cases depending on the value of w_0 when the algorithm terminated. Suppose first that when the algorithm terminated w_0 was *true*. In this case we will additionally prove that for $1 \leq k < n$ D_k is an analytic submanifold of \mathbb{R}^k , all elements of W_k are order-invariant in D_k , and if $k < n - 1$ then none of the elements of W_{k+1} vanishes identically at any point in D_k , W_{k+1} is analytic delineable on D_k , and the elements of W_{k+1} are order-invariant in each W_{k+1} -section over D_k . If a_1 is a root of an element of W_1 let $D_1 = \{a_1\}$ else let $D_1 = (r_1, s_1)$, where r_1 and s_1 are roots of elements of W_1 , $-\infty$, or ∞ , $r_1 < a_1 < s_1$, and there are no roots of W_1 in (r_1, s_1) . D_1 is a connected analytic submanifold of \mathbb{R}^1 and all elements of W_1 are order-invariant in D_1 . Since the elements of W_2 are irreducible, none of the elements of W_2 vanishes identically at any point in D_1 . Since all irreducible factors of elements of $LProjMC(W_2, \Pi_1(a))$ belong to W_1 , by Lemma 15, W_2 is analytic delineable over D_1 and the elements of W_2 are order-invariant in each W_2 -section over D_1 . Suppose that, for some $1 < k < n - 1$, we have constructed D_{k-1} satisfying the required conditions. The conditions imply that W_k is analytic delineable on D_{k-1} . Let D_k be the W_k -section or W_k -sector over D_{k-1} which contains $\Pi_k(a)$. D_k is an analytic submanifold of \mathbb{R}^k . The elements of W_k are order-invariant in D_k , because they are order-invariant in each W_k -section over D_{k-1} and nonzero in each W_k -sector over D_{k-1} . Since all irreducible factors of elements of $LProjMC(W_{k+1}, \Pi_k(a))$ belong to $W_1 \cup \dots \cup W_k$, by Lemma 15, W_{k+1}^* is analytic delineable over D_k and the elements of W_{k+1}^* are order-invariant in each W_{k+1}^* -section over D_k . Step (2b) guarantees that if $k < n - 1$ then $W_{k+1}^* = W_{k+1}$.

Suppose now that when the algorithm terminated w_0 was *false*. Let D_1 be as in the first part of the proof. As before, W_2 is analytic delineable over D_1 and the elements of W_2 are order-invariant in each W_2 -section over D_1 . Let D_2 be the W_2 -section or W_2 -sector over D_1 which contains (a_1, a_2) . D_2 is an analytic submanifold of \mathbb{R}^2 . The elements of W_2 are order-invariant in D_2 , because they are order-invariant in each W_2 -section over D_1 and nonzero in each W_2 -sector over D_1 . Since all irreducible factors of elements of $LProjMC(W_3, \Pi_2(a))$ belong to $W_1 \cup W_2$, by Lemma 15, W_3^* is analytic delineable over D_2 . Suppose that, for some $2 < k < n - 1$, we have constructed D_{k-1} satisfying the required conditions. The conditions on D_{k-1} imply that W_k^* is delineable on D_{k-1} . Let D_k be the W_k^* -section or

W_k^* -sector over D_{k-1} which contains $\Pi_k(a)$. All elements of W_k are sign-invariant in D_k . Since all irreducible factors of elements of $LProjH(W_{k+1}, \Pi_k(a))$ belong to $W_1 \cup \dots \cup W_k$, by Lemma 16, W_{k+1}^* is delineable over D_k .

Since for $1 \leq k < n$, D_k is the W_k^* -section or W_k^* -sector over D_{k-1} which contains $\Pi_k(a)$, D_k is the maximal connected set containing $\Pi_k(a)$ such that all elements of W_j for $1 \leq j \leq k$ have constant signs on D_j . \square

Correctness and termination of Algorithm 12 is obvious.

Proposition 18. *Algorithm 13 terminates and the returned pair (F, V) satisfies the required conditions.*

Proof. Let P_S be the set of all polynomials that appear in S and let $W_H = (W_{H,1}, \dots, W_{H,n})$ be the Hong's projection sequence [11] for P_S (the variant of given in Proposition 7 of [23]). Suppose that $\bar{P} \subseteq W_{H,1} \cup \dots \cup W_{H,k+1}$ and $\bar{a} \in \mathbb{R}^k$, where $k < n$. Let $(W_1, \dots, W_{k+1}) = LocalProjection(P, \bar{a})$. Since we assume that finite sets of polynomials are consistently ordered according to a fixed linear order in the set of all polynomials, $W_i \subseteq W_{H,i}$ for $1 \leq i \leq k+1$. Hence all polynomials that appear during execution of *LPCAD* are elements of $W_{H,1} \cup \dots \cup W_{H,n}$. In particular, r_1 and r_2 that appear in the elements of *stack* are roots of elements of $W_{H,k+1}$, $-\infty$, or ∞ . Therefore, the number of possible elements of *stack* is finite, and hence the loop in step (3) terminates. Recursive calls to *TDCAD* increment k . When $k = n - 1$ then either step (3c) yields $H = false$ or step (3d) yields $H = true$, and hence step (3e) containing the recursive call to *LPCAD* is never executed. Therefore the value of k is bounded by $n - 1$, and hence the recursion terminates.

Let (F, V) be the pair returned by *LPCAD* and suppose that $C \subseteq \mathbb{R}^k$ is a cell such that $\bar{a} \in C$ and for $1 \leq j \leq k$ all elements of V_j have constant signs on $\Pi_j(C)$. We need to show that

$$(x_1, \dots, x_k) \in C \Rightarrow (F(x_1, \dots, x_n) \iff S(x_1, \dots, x_n))$$

Let $c = (c_1, \dots, c_n) \in \mathbb{R}^n$ and $\bar{c} = (c_1, \dots, c_k) \in C$. We need to show that $F(c) = S(c)$. Let $W = LocalProjection(Q, \bar{a})$, as computed in step (5). All elements of W_j have constant signs on $\Pi_j(C)$, for $1 \leq j \leq k$. Since none of the elements of Q vanishes identically at \bar{a} , Q is delineable over C . Hence the Q -sections and the Q -sectors over C form a partition of $C \times \mathbb{R}$.

For a tuple $\theta = (u_1, r_1, \rho_1, u_2, r_2, \rho_2)$ that appears on *stack* in any iteration of the loop in step (3) put

$$Z_1(\theta) = \{(\bar{x}, x_{k+1}) \in \mathbb{R}^{k+1} : \bar{x} \in C \wedge r_1 \rho_1 x_{k+1} \rho_2 r_2\}$$

For each $\alpha = (a_{k+1}, G \wedge H) \in A$ put

$$Z_2(\alpha) = \{(\bar{x}, x_{k+1}) \in \mathbb{R}^{k+1} : \bar{x} \in C \wedge G(\bar{x}, x_{k+1})\}$$

Note that each $Z_1(\theta)$ and $Z_2(\alpha)$ is a union of Q -sections and Q -sectors over C . Put $\Omega_1 = \{Z_1(\theta) : \theta \in stack\}$ and $\Omega_2 = \{Z_2(\alpha) : \alpha \in A\}$. We will show that in each instance of the loop in step (3) $\Omega_1 \cup \Omega_2$ is a partition of $C \times \mathbb{R}$. In the first instance of the loop in step (3) $\Omega_1 = \{C \times \mathbb{R}\}$ and $\Omega_2 = \emptyset$, and hence $\Omega_1 \cup \Omega_2$ is a partition of $C \times \mathbb{R}$. We will show that this property is preserved in each instance of the loop. In each instance a tuple $\theta = (u_1, r_1, \rho_1, u_2, r_2, \rho_2)$ is removed from *stack* and $\alpha = (a_{k+1}, G \wedge H)$ is added to A . If $u_1 = u_2$ in step (3g) then $Z_2(\alpha) = Z_1(\theta)$ and the property is preserved. If $v_1 = v_2$ in step (3j) then $G = (x_{k+1} = s_1)$ and tuples $\theta_2 = (v_1, s_1, <, u_2, r_2, \rho_2)$ and $\theta_1 = (u_1, r_1, \rho_1, v_1, s_1, <)$ are added to *stack*. Since $\{Z_1(\theta_1), Z_2(\alpha), Z_1(\theta_2)\}$ is a partition of $Z_1(\theta)$, the property is preserved. Otherwise steps (3k)-(3m) are executed. If in step (3k) $u_2 > v_2$ or $u_2 = v_2$ and $\rho_2 = \leq$ then put $Z_{1,2} = Z_1(\theta_2)$, where $\theta_2 = (v_2, s_2, \leq, u_2, r_2, \rho_2)$ is the tuple added

to *stack*, else put $Z_{1,2} = \emptyset$. If in step (3l) $v_1 > u_1$ or $v_1 = u_1$ and $\rho_1 = \leq$ then put $Z_{1,1} = Z_1(\theta_1)$, where $\theta_1 = (u_1, r_1, \rho_1, v_1, s_1, \leq)$ is the tuple added to *stack*, else put $Z_{1,1} = \emptyset$. Since $\{Z_{1,1}, Z_2(\alpha), Z_{1,2}\}$ is a partition of $Z_1(\theta)$, the property is preserved.

After the loop in step (3) is finished *stack* is empty, $\Omega_1 = \emptyset$, and hence Ω_2 is a partition of $C \times \mathbb{R}$. Let $\alpha = (a_{k+1}, G \wedge H) \in A$ be such that $(\bar{c}, c_{k+1}) \in Z_2(\alpha)$. Let us analyze the instance of the loop in step (3) which resulted in adding α to A . Let $D = Z_2(\alpha)$.

Suppose first that $H = \text{false}$ or $H = \text{true}$ was found in step (3c) or (3d). Let $W = \text{LocalProjection}(P \cup R, \bar{a})$, as computed in step (3c) or (3d). For $1 \leq j \leq k$, $W_j \subseteq V_j$, and hence all elements of W_j have constant signs on $\Pi_j(D)$. Therefore the set W_{k+1}^* of elements of W_{k+1} that are not identically zero on $C \times \mathbb{R}$ is delineable over C . By definition of G , D is a W_{k+1}^* -section or a W_{k+1}^* -sector over C . Hence all elements of W_{k+1} have constant signs on D . In particular, all elements of P have constant signs on D , and so $S(c) = H = F(c)$.

Now suppose that $(H, U) = \text{LPCAD}(S, \bar{b})$ was computed in step (3e). Let

$$W = \text{LocalProjection}(U_{k+1} \cup R, \bar{a})$$

For $1 \leq j \leq k$, $W_j \subseteq V_j$, and hence all elements of W_j have constant signs on $\Pi_j(D)$. As before, W_{k+1}^* is delineable over C , D is a W_{k+1}^* -section or a W_{k+1}^* -sector over C , and all elements of W_{k+1} have constant signs on D . In particular, all elements of U_{k+1} have constant signs on D . Since for $1 \leq j \leq k$ $U_j \subseteq V_j$, all elements of U_j have constant signs on $\Pi_j(D)$. Hence

$$(x_1, \dots, x_k, x_{k+1}) \in D \Rightarrow (H(x_1, \dots, x_n) \iff S(x_1, \dots, x_n))$$

and so $F(c) = H(c) = S(c)$. \square

3.4. Implementation remarks.

Remark 19. The following somewhat technical improvements have been observed to improve practical performance of Algorithm 13.

- (1) In step (2c) of Algorithm 8 in q_l may be chosen arbitrarily as long as $q_l(\bar{a}) \neq 0$, hence an implementation may choose the simplest q_l .
- (2) If in a recursive call to $\text{LPCAD}(S, (a_1, \dots, a_k))$ the initial coordinates (a_1, \dots, a_m) correspond to single-point intervals, that is $u_1 = u_2$ in step (3b) of the currently evaluated iteration of loop (3) in all parent computations of

$$\text{LPCAD}(S, (a_1, \dots, a_j))$$

for $1 \leq j \leq m$, then $\text{LocalProjection}(P, (a_1, \dots, a_k))$ does not need to compute the last m levels of projection. Instead it can return $W = (W_1, \dots, W_n)$ with $W_1 = \dots = W_m = \emptyset$.

- (3) Computations involved in finding projections are repeated multiple times. A practical implementation needs to make extensive use of caching.

3.5. Example.

In this section we apply LPCAD to solve the problem stated in Example 1.

In step (1) of $\text{LPCAD}(S, ())$ we compute $S_{\text{CNF}} = (f_1 < 0 \vee f_2 \leq 0) \wedge (f_1 < 0 \vee f_3 \leq 0)$ and $S_{\text{DNF}} = f_1 < 0 \vee (f_2 \leq 0 \wedge f_3 \leq 0)$. In the first iteration of loop (3) we remove a tuple representing $-\infty < x < \infty$ from *stack* and pick $a_1 = 0$. The calls to PEval in steps (3c) and (3d) yield *undecided*. Step (3e) makes a recursive call to $\text{LPCAD}(S, (0))$.

In the first iteration of loop (3) in $\text{LPCAD}(S, (0))$ we remove a tuple representing $-\infty < y < \infty$ from *stack* and pick $a_2 = 0$. $\text{PEval}(S_{\text{CNF}}, (0, 0))$ in step (3c) yields $(\text{true}, \{f_1, f_2, f_3\})$.

We continue on to step (3d) where $PEval(S_{DNF}, (0, 0))$ yields $(true, \{f_1\})$. We set $H = true$ and compute

$$W = LocalProjection(\{f_1\}, (0)) = (W_1, \{f_1\})$$

where $W_1 = \{x-1, x+1\}$ is the set of factors of $discr_y f_1 = 16(x^2 - 1)$. We go to step (3f) and set $V_1 = V_1 \cup W_1 = \{x-1, x+1\}$. In step (3h) we find $s_1 = Root_{y,1} f_1 = -2\sqrt{1-x^2}$, $s_2 = Root_{y,2} f_1 = 2\sqrt{1-x^2}$, $v_1 = -2$, and $v_2 = 2$. In step (3i) we set $Q = Q \cup \{f_1\} = \{f_1\}$. In steps (3k) and (3l) we add tuples representing $2 \leq y < \infty$ and $-\infty < y \leq -2$ to *stack*. In step (3n) we obtain $A = \{(0, -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2})\}$.

In the second iteration of loop (3) in $LPCAD(S, (0))$ we remove a tuple representing $-\infty < y \leq -2$ from *stack* and pick $a_2 = -4$. $PEval(S_{CNF}, (0, -4))$ in step (3c) yields $(false, \{f_1, f_2\})$. We set $H = false$ and compute

$$W = LocalProjection(\{f_1, f_2\}, (0)) = (W_1, \{f_1, f_2\})$$

where $W_1 = \{x-1, x+1\}$ is the set of factors of $discr_y f_1 = 16(x^2 - 1)$, $discr_y f_2 = 4(x^2 - 1)$, and $res_y(f_1, f_2) = 9(x^2 - 1)^2$. We go to step (3f) and set $V_1 = V_1 \cup W_1 = \{x-1, x+1\}$. In step (3h) we find $s_1 = v_1 = -\infty$, $s_2 = Root_{y,1} f_1 = -2\sqrt{1-x^2}$, and $v_2 = -2$. In step (3i) we set $Q = Q \cup \{f_1\} = \{f_1\}$. In step (3k) we add a tuple representing $y = -2$ to *stack*. In step (3n) we obtain $A = \{(0, -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}), (-4, false)\}$.

In the third iteration of loop (3) in $LPCAD(S, (0))$ we remove a tuple representing $y = -2$ from *stack* and set $a_2 = -2$. $PEval(S_{CNF}, (0, -2))$ in step (3c) yields

$$(false, \{f_1, f_2\})$$

We set $H = false$ and compute

$$W = LocalProjection(\{f_1, f_2\}, (0)) = (W_1, \{f_1, f_2\})$$

where $W_1 = \{x-1, x+1\}$. We go to step (3f) and set $V_1 = V_1 \cup W_1 = \{x-1, x+1\}$. In step (3g) we set $G = (y = -2\sqrt{1-x^2})$. In step (3n) we obtain $A = \{(0, -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}), (-4, false), (-2, false)\}$.

The remaining two iterations of loop (3) look very similar to the last two. In step (4) we obtain $F = -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}$. In step (5) we compute

$$W = LocalProjection(\{f_1\}, (0)) = (\{x-1, x+1\}, \{f_1\})$$

and in step (6) we set $V_1 = V_1 \cup W_1 = \{x-1, x+1\}$. The returned value is $(-2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}, (\{x-1, x+1\}))$.

In step (3e) of $LPCAD(S, (0))$ we obtain $H = -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2}$ and $U = (\{x-1, x+1\})$.

$$LocalProjection(\{x-1, x+1\}, (0))$$

yields $(\{x-1, x+1\})$. In step (3h) we find $s_1 = Root_{x,1}(x+1) = -1$, $s_2 = Root_{x,1}(x-1) = 1$, $v_1 = -1$, and $v_2 = 1$. In steps (3k) and (3l) we add tuples representing $1 \leq x < \infty$ and $-\infty < x \leq -1$ to *stack*. In step (3n) we obtain $A = \{(0, -1 < x < 1 \wedge -2\sqrt{1-x^2} < y < 2\sqrt{1-x^2})\}$.

In the second iteration of loop (3) in $LPCAD(S, (0))$ we remove a tuple representing $-\infty < x \leq -1$ from *stack* and pick $a_1 = -2$. The calls to $PEval$ in steps (3c) and (3d) yield *undecided*. Step (3e) makes a recursive call to $LPCAD(S, (-2))$.

In the first iteration of loop (3) in $LPCAD(S, (-2))$ we remove a tuple representing $-\infty < y < \infty$ from *stack* and pick $a_2 = 0$. $PEval(S_{CNF}, (-2, 0))$ in step (3c) yields

$$(false, \{f_1, f_2\})$$

We set $H = \text{false}$ and compute

$$W = \text{LocalProjection}(\{f_1, f_2\}, (-2)) = (W_1, \{f_1, f_2\})$$

where $W_1 = \{x - 1, x + 1\}$ is the set of factors of $\text{discr}_y f_1$ and $\text{discr}_y f_2$ ($\text{res}_y(f_1, f_2)$ is not a part of the projection because $f_1(-2, y)$ and $f_2(-2, y)$ have no real roots). We go to step (3f) and set $V_1 = V_1 \cup W_1 = \{x - 1, x + 1\}$. In step (3h) we find $s_1 = v_1 = -\infty$ and $s_2 = v_2 = \infty$. In step (3i) Q remains empty. In step (3n) we obtain $A = \{(0, \text{false})\}$. The loop ends after one iteration and the returned value is $(\text{false}, \{x - 1, x + 1\})$.

In step (3e) of $\text{LPCAD}(S, ())$ we obtain $H = \text{false}$ and $U = (\{x - 1, x + 1\})$.

$$\text{LocalProjection}(\{x - 1, x + 1\}, ())$$

yields $(\{x - 1, x + 1\})$. In step (3h) we find $s_1 = v_1 = -\infty$, $s_2 = \text{Root}_{x,1}(x + 1) = -1$, and $v_2 = -1$. In step (3k) we add a tuple representing $x = -1$ to *stack*. In step (3n) we obtain $A = \{(0, -1 < x < 1 \wedge -2\sqrt{1 - x^2} < y < 2\sqrt{1 - x^2}), (-2, \text{false})\}$.

In the third iteration of loop (3) in $\text{LPCAD}(S, ())$ we remove a tuple representing $x = -1$ from *stack* and pick $a_1 = -2$. The calls to PEval in steps (3c) and (3d) yield *undecided*. Step (3e) makes a recursive call to $\text{LPCAD}(S, (-1))$.

In the first iteration of loop (3) in $\text{LPCAD}(S, (-1))$ we remove a tuple representing $-\infty < y < \infty$ from *stack* and pick $a_2 = 0$. $\text{PEval}(S_{\text{CNF}}, (-1, 0))$ in step (3c) yields

$$(\text{false}, \{f_1, f_3\})$$

We set $H = \text{false}$ and compute

$$W = \text{LocalProjection}(\{f_1, f_3\}, (-1)) = (W_1, \{f_1, f_2\})$$

where, by Remark 19, we can take $W_1 = \emptyset$. We go to step (3f) and the set V_1 remains empty. In step (3h) we find $s_1 = s_2 = \text{Root}_{y,1} f_1$ and $v_1 = v_2 = 0$. In step (3i) we set $Q = Q \cup \{f_1\} = \{f_1\}$. In step (3j) we add tuples representing $0 < x < \infty$ and $-\infty < x < 0$ to *stack*. In step (3n) we obtain $A = \{(0, \text{false})\}$.

In the second iteration of loop (3) in $\text{LPCAD}(S, (-1))$ we remove a tuple representing $-\infty < y < 0$ from *stack* and pick $a_2 = -1$. $\text{PEval}(S_{\text{CNF}}, (-1, -1))$ in step (3c) yields $(\text{false}, \{f_1, f_2\})$. We set $H = \text{false}$ and compute $W = \text{LocalProjection}(\{f_1, f_2\}, (-1)) = (W_1, \{f_1, f_2\})$, where, by Remark 19, we can take $W_1 = \emptyset$. We go to step (3f) and the set V_1 remains empty. In step (3h) we find $s_1 = v_1 = -\infty$, $s_2 = \text{Root}_{y,1} f_1$ and $v_2 = 0$. In step (3i) we set $Q = Q \cup \{f_1\} = \{f_1\}$. In step (3n) we obtain $A = \{(0, \text{false}), (-1, \text{false})\}$.

The remaining iteration of loop (3) look very similar to the last one. In step (4) we obtain $F = \text{false}$. In step (5) we compute

$$W = \text{LocalProjection}(\{f_1\}, (-1)) = (\emptyset, \{f_1\})$$

by Remark 19. The returned value is $(\text{false}, (\emptyset))$.

In step (3e) of $\text{LPCAD}(S, ())$ we obtain $H = \text{false}$ and $U = (\emptyset)$. $\text{LocalProjection}(\emptyset, ())$ yields (\emptyset) . In step (3g) we set $G = (x = -1)$. In step (3n) we obtain $A = \{(0, -1 < x < 1 \wedge -2\sqrt{1 - x^2} < y < 2\sqrt{1 - x^2}), (-2, \text{false}), (-1, \text{false})\}$.

The remaining two iterations of loop (3) look very similar to the last two. In step (4) we obtain $F = -1 < x < 1 \wedge -2\sqrt{1 - x^2} < y < 2\sqrt{1 - x^2}$ and the returned value is $(-1 < x < 1 \wedge -2\sqrt{1 - x^2} < y < 2\sqrt{1 - x^2}, ())$.

4. EMPIRICAL RESULTS

Algorithm 13 (*LPCAD*) and the cylindrical algebraic decomposition (*CAD*) algorithm have been implemented in C, as a part of the kernel of *Mathematica*. The experiments have been conducted on a Linux server with a 32-core 2.4 GHz Intel Xeon processor and 378 GB of RAM available for all processes. The reported CPU time is a total from all cores used. Since we do not describe the use of equational constraints in the current paper, we have selected examples that do not involve equations.

4.1. Benchmark examples. We compare the performance of *LPCAD* and *CAD* for the following three problems and for the 7 examples from Wilson's benchmark set [26] (version 4) that do not contain equations.

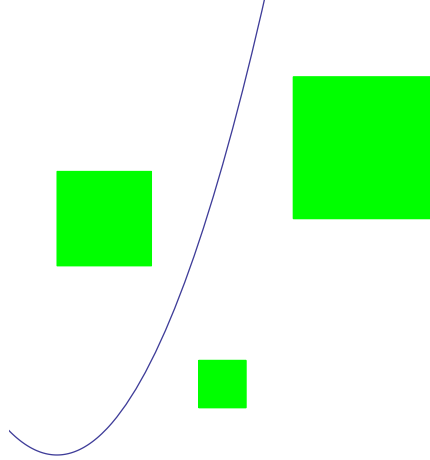
Example 20. (*Two quadratics*) Find a cylindrical algebraic decomposition of the solution set of $ax^2 + bx + c \geq 0 \wedge dx^2 + ex + f \geq 0$ with the variables ordered (a, b, c, d, e, f, x) .

Example 21. (*Ellipse in a square*) Find conditions for ellipse $\frac{(x-c)^2}{a} + \frac{(y-d)^2}{b} < 1$ to be contained in the square $-1 < x < 1 \wedge -1 < y < 1$. We compute a cylindrical algebraic decomposition of the solution set of

$$\forall x, y \in \mathbb{R} \ a > 0 \wedge b > 0 \wedge b(x-c)^2 + a(y-d)^2 < ab \Rightarrow \\ -1 < x < 1 \wedge -1 < y < 1$$

with the free variables ordered (a, b, c, d) .

Example 22. (*Distance to three squares*) Find the distance of a point on the parabola shown in the picture to the union of three squares.



We compute a cylindrical algebraic decomposition of the solution set of

$$\exists x, y \in \mathbb{R} \ (x-a)^2 + (y-a^2+2)^2 \leq d \wedge \\ (0 \leq x \leq 1 \wedge 0 \leq y \leq 1 \vee \\ \frac{3}{2} \leq x \leq 2 \wedge -\frac{3}{2} \leq y \leq -1 \vee \\ \frac{5}{2} \leq x \leq 4 \wedge \frac{1}{2} \leq y \leq 2)$$

with the free variables ordered (a, d) .

TABLE 1. Benchmark examples

Example	Time		Cells		WO
	<i>CAD</i>	<i>LPCAD</i>	<i>CAD</i>	<i>LPCAD</i>	
20	97.7	2.61	324137	3971	N
21	> 100000	38.1	?	67535	N
22	2402	44.9	13105366	71411	Y
W 2.3	0.063	0.088	91	84	Y
W 2.8	0.015	0.015	15	15	Y
W 2.9	0.047	0.011	59	19	Y
W 2.10	0.135	0.197	779	647	Y
W 2.11	0.045	0.007	463	31	N
W 2.16	0.076	0.025	644	4	Y
W 6.5	2.10	1.58	11279	2536	Y

TABLE 2. Randomly generated examples

Var No.	Time			Cells			TO	WO
	<i>CAD/LPCAD</i>			<i>CAD/LPCAD</i>				
	Mean	Min	Max	Mean	Min	Max		
5	1.64	0.50	11.1	2.55	0.75	17.3	8	4
6	3.82	0.80	55.7	6.14	1	98.4	1	10
7	26.9	5.10	257	43.2	6.74	408	3	0

Results of experiments are given in Table 1. Examples from [26] are marked with W and the original number. The columns marked Time give the CPU time, in seconds, used by each algorithm. The columns marked Cells give the number of cells constructed by each algorithm. The column marked WO tells whether the system is well-oriented.

4.2. Randomly generated examples. For this experiment we used randomly generated systems with 5, 6, and 7 variables, 25 systems with each number of variables. The systems had the form $f < 0$ or $f \leq 0$, with a quadratic polynomial f with 6 to 15 terms and 10-bit integer coefficients. We selected systems for which at least one of the algorithms finished in 1000 seconds. Results of experiments are given in Table 2. The columns marked Time give the ratio of *CAD* timing divided by *LPCAD* timing. The columns marked Cells give the ratio of the numbers of cells constructed by *CAD* and by *LPCAD*. The ratios are computed for the examples for which both algorithms finished in 1000 seconds. The columns marked Mean give geometric means. The column marked TO gives the number of examples for which *CAD* did not finish in 1000 seconds. *LPCAD* finished in 1000 seconds for all examples. The column marked WO gives the number of systems that were well-oriented.

4.3. Conclusions. Experiments suggest that for systems that are not well-oriented *LPCAD* performs better than *CAD*. For well oriented-systems *LPCAD* usually construct less cells than *CAD*, but this does not necessarily translate to a faster timing, due to overhead from re-constructing projection for every cell. However, for some of the well-oriented systems, for instance Example 22, *LPCAD* is significantly faster than *CAD*, due to its ability to exploit the Boolean structure of the problem. Unfortunately we do not have a precise characterisation of such problems. Nevertheless *LPCAD* may be useful for well-oriented

problems that prove hard for the CAD algorithm or may be tried in parallel with the CAD algorithm.

REFERENCES

- [1] S. Basu, R. Pollack, and M. Roy. *Algorithms in real algebraic geometry*, volume 10. Springer-Verlag New York Inc, 2006.
- [2] C. W. Brown. Improved projection for cylindrical algebraic decomposition. *J. Symbolic Comp.*, 32:447–465, 2001.
- [3] C. W. Brown. Qepcad b - a program for computing with semi-algebraic sets using cads. *ACM SIGSAM Bulletin*, 37:97–108, 2003.
- [4] C. W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2013*, pages 133–140. ACM, 2013.
- [5] B. Caviness and J. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, New York, 1998. Springer Verlag.
- [6] C. Chen, M. M. Maza, B. Xia, and L. Yang. Computing cylindrical algebraic decomposition via triangular decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2009*, pages 95–102. ACM, 2009.
- [7] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. *Lect. Notes Comput. Sci.*, 33:134–183, 1975.
- [8] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symbolic Comp.*, 12:299–328, 1991.
- [9] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. In *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, 1998.
- [10] D. Grigoriev and N. Vorobjov. Solving systems of polynomial inequalities in subexponential time. *J. Symb. Comput.*, 5(1/2):37–64, 1988.
- [11] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 1990*, pages 261–264. ACM, 1990.
- [12] H. Hong and M. S. E. Din. Variant quantifier elimination. *J. Symb. Comput.*, 47:883–901, 2012.
- [13] D. Jovanovic and L. M. de Moura. Solving non-linear arithmetic. In *IJCAR*, pages 339–354, 2012.
- [14] S. Łojasiewicz. *Ensembles semi-analytiques*. I.H.E.S., 1964.
- [15] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993.
- [16] S. McCallum. An improved projection for cylindrical algebraic decomposition of three dimensional space. *J. Symbolic Comp.*, 5:141–161, 1988.
- [17] S. McCallum. An improved projection for cylindrical algebraic decomposition. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer Verlag, 1998.
- [18] J. Renegar. On the computational complexity and geometry of the first order theory of the reals. *J. Symbolic Comp.*, 13:255–352, 1992.
- [19] A. Strzeboński. Computing in the field of complex algebraic numbers. *J. Symbolic Comp.*, 24:647–656, 1997.
- [20] A. Strzeboński. Solving systems of strict polynomial inequalities. *J. Symbolic Comp.*, 29:471–480, 2000.
- [21] A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *J. Symbolic Comp.*, 41:1021–1038, 2006.
- [22] A. Strzeboński. Computation with semialgebraic sets represented by cylindrical algebraic formulas. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2010*, pages 61–68. ACM, 2010.
- [23] A. Strzeboński. Solving polynomial systems over semialgebraic sets represented by cylindrical algebraic formulas. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2012*, pages 335–342. ACM, 2012.
- [24] A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, 1951.
- [25] V. Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *AAECC*, 8:85–101, 1993.
- [26] D. Wilson. Real geometry and connectedness via triangular description: Cad example bank, 2012. <http://opus.bath.ac.uk/29503/>.

WOLFRAM RESEARCH INC., 100 TRADE CENTRE DRIVE, CHAMPAIGN, IL 61820, U.S.A.
E-mail address: `adams@wolfram.com`